AMENDMENTS TO THE SPECIFICATION:

Please amend the specification as follows:

Please amend the paragraph on page 13, lines 7-12, of Applicants' specification as follows:

As mentioned above, the development process in system 100 is based on a metamodel. FIG. 2 illustrates a metamodel200 (e.g., a semantic infonnation model), which is represented using a unified modeling language (UML) class diagram, in conjunction with a portion 205 of a development process for developing a user interface application 260. The specifications for UML~~, which~~ are set forth by the Object Management Group (OMG)~~, can be found on OMG's web site at http://www.omg.org/umll~~.

Please amend the paragraph at page 29, line 19 to page 30, line 15 of Applicants' specification as follows:

FIG. 11 illustrates a sample process 1100 for generating an API (e.g., metadata API 130 of system 100) and/or an XML schema from a metamodel. The metamodel in this example is represented using UML, and has the standard and customizable constructs described above. In process 1100, a metamodel administrator 1105 uses a modeling tool 1110 to create and/or modify a UML class diagram 1115 representing the metamodel (e.g., metamodel 200, metamodel 300, or the metamodel portions in FIGS. 5-10) that serves as the basis for the derivation. Once a metamodel is complete, process 1100 generates a standards-based XML Metadata Interchange (XMI) model description 1120. That is, XMI model 1120 is a representation ofthe UML metamodel1115, written in XML and described according to the XMI standard. Process 1100 parses the XMI model description 1120 using a Simple API for XML (SAX)-based XML parser 1125 to generate a representation of the meta model using a set of intermediate objects 1130 (e.g., Java objects). ~~(The SAX technical specification can be found, for example, at http://www.saxproject.org.)~~ The Java objects 1130 represent the document object model (DOM) of the metamodel (e.g., represented as a 10 DOM tree). Process 1100 uses Java objects 1130 as inputs to generators 1135, 1140, and 1145 to generate code that is included in metadata API 130. As described in more detail below, metadata API 130 includes a portion 130a that includes interfaces, proxies, and state classes, a portion 130b that includes XML marshalling code, and a portion 130c that includesan XML schema. Process 1100 also uses Java code templates 1155 and schema templates 1160 in the generation process to generate metadata API 130.